Contents

Dry Run

4 minute read page test

This task shows you how to set up an Istio authorization policy using a new experimental annotation `istio.io/dry-run` to dry-run the policy without actually enforcing it.

The dry-run annotation allows you to better understand the effect of an authorization policy

before applying it to the production traffic. This helps to reduce the risk of breaking the production traffic caused by an incorrect authorization policy.

The following information describes an experimental feature, which is intended for evaluation purposes only.

Before you begin

Before you begin this task, do the following:

- Read the Istio authorization concepts.
- Follow the Istio installation guide to install Istio.
- Deploy Zipkin for checking dry-run tracing results. Follow the Zipkin task to install Zipkin in the cluster. Make sure the sampling rate is set to 100 which allows you to quickly reproduce the trace span in the task.
- Deploy Prometheus for checking dry-run metric results. Follow the Prometheus task to install the Prometheus in the cluster.

- Deploy test workloads:

  This task uses two workloads, `httpbin` and `sleep`, both deployed in namespace `foo`. Both workloads run with an Envoy proxy sidecar. Create the `foo` namespace and deploy the workloads with the following command:

  ```
  $ kubectl create ns foo
  $ kubectl label ns foo istio-injection=enabled
  $ kubectl apply -f @samples/httpbin/httpbin.yaml@ -n foo
  $ kubectl apply -f @samples/sleep/sleep.yaml@ -n foo
  ```

- Enable proxy debug level log for checking dry-run logging results:

```
$ istioctl proxy-config log deploy/httpbin.foo --level "rba
c:debug" | grep rbac
rbac: debug
```

- Verify that `sleep` can access `httpbin` with the following command:

```
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o js
onpath={.items..metadata.name})" -c sleep -n foo -- curl ht
tp://httpbin.foo:8000/ip -s -o /dev/null -w "%{http_code}\n
"
200
```

If you don’t see the expected output as you follow the task, retry after a few seconds.

Caching and propagation overhead can cause some delay.

Create dry-run policy

1. Create an authorization policy with dry-run annotation `"istio.io/dry-run": "true"` with the following command:

```
$ kubectl apply -n foo -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: deny-path-headers
  annotations:
    "istio.io/dry-run": "true"
spec:
  selector:
    matchLabels:
      app: httpbin
  action: DENY
  rules:
  - to:
    - operation:
        paths: ["/headers"]
EOF
```

2. Verify a request to path `/headers` is allowed

because the policy is created in dry-run mode:

```
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o js
onpath={.items..metadata.name})" -c sleep -n foo -- curl "h
ttp://httpbin.foo:8000/headers" -s
```

Check dry-run result in proxy log

1. The dry-run results can be found in the proxy debug log, similar to `shadow denied, matched policy ns[foo]-policy[deny-path-headers]-rule[0]`. See the

troubleshooting guide for more details.

Check dry-run result in metric using Prometheus

1. Open the Prometheus dashboard with the following command:

```
$ istioctl dashboard prometheus
```

2. In the Prometheus dashboard, search for the

metric `envoy_http_inbound_0_0_0_0_80_rbac{authz_dry_run_result!=""}`. The following is an example metric output:

```
envoy_http_inbound_0_0_0_0_80_rbac{app="httpbin",authz_dry_
run_action="deny",authz_dry_run_result="allowed",instance="
10.44.1.11:15020",istio_io_rev="default",job="kubernetes-po
ds",kubernetes_namespace="foo",kubernetes_pod_name="httpbin
-74fb669cc6-95qm8",pod_template_hash="74fb669cc6",security_
istio_io_tlsMode="istio",service_istio_io_canonical_name="h
ttpbin",service_istio_io_canonical_revision="v1",version="v
1"} 0
envoy_http_inbound_0_0_0_0_80_rbac{app="httpbin",authz_dry_
run_action="deny",authz_dry_run_result="denied",instance="1
0.44.1.11:15020",istio_io_rev="default",job="kubernetes-pod
s",kubernetes_namespace="foo",kubernetes_pod_name="httpbin-
74fb669cc6-95qm8",pod_template_hash="74fb669cc6",security_i
stio_io_tlsMode="istio",service_istio_io_canonical_name="ht
tpbin",service_istio_io_canonical_revision="v1",version="v1
"}  1
```

3. The metric

   `envoy_http_inbound_0_0_0_0_80_rbac{authz_dry_run_res`

`ult="denied"}` has value `1` (you might find different value depending on how many requests you have sent. It's expected as long as the value is greater than 0). This means the dry-run policy applied to the `httpbin` workload on port `80` matched one request. The policy would reject the request once if it was not in dry-run mode.

Check dry-run result in tracing using Zipkin

1. Open the Zipkin dashboard with the following command:

```
$ istioctl dashboard zipkin
```

2. Find the trace result for the request from `sleep` to `httpbin`. Try to send some more requests if you do see the trace result due to the delay in the Zipkin.

3. In the trace result, you should find the following custom tags indicating the request is rejected by the dry-run policy `deny-path-headers` in the namespace `foo`:

```
istio.authorization.dry_run.deny_policy.name: ns[foo]-policy[deny-path-headers]-rule[0]
istio.authorization.dry_run.deny_policy.result: denied
```

Summary

The Proxy debug log, Prometheus metric and Zipkin trace results indicate that the dry-run policy will reject the request. You can further change the policy if the dry-run result is not expected.

It’s recommended to keep the dry-run policy for some

additional time so that it can be tested with more production traffic.

When you are confident about the dry-run result, you can disable the dry-run mode so that the policy will start to actually reject requests. This can be achieved by either of the following approaches:

- Remove the dry-run annotation completely; or
- Change the value of the dry-run annotation to `false`.

Limitations

The dry-run annotation is currently in experimental stage and has the following limitations:

- The dry-run annotation currently only supports ALLOW and DENY policies;
- There will be two separate dry-run results (i.e. log, metric and tracing tag) for ALLOW and DENY policies due to the fact that the ALLOW and DENY policies are enforced separately in the proxy. You should take all the two dry-run results into consideration because a request could be

allowed by an ALLOW policy but still rejected by another DENY policy;

- The dry-run results in the proxy log, metric and tracing are for manual troubleshooting purposes and should not be used as an API because it may change anytime without prior notice.

Clean up

1. Remove the namespace `foo` from your configuration:

```
$ kubectl delete namespace foo
```

2. Remove Prometheus and Zipkin if no longer needed.